А.Р. Магомедов [1], *А.В. Горобец* [2]

ГЕТЕРОГЕННАЯ РЕАЛИЗАЦИЯ ПРЕДОБУСЛАВЛИВАТЕЛЕЙ НА ОСНОВЕ МЕТОДА ГАУССА–ЗЕЙДЕЛЯ ДЛЯ РАЗРЕЖЕННОЙ БЛОЧНОЙ МАТРИЦЫ*

Введение

Задачей данной работы является создание гетерогенного параллельного предобуславливателя для решателя системы линейных алгебраических уравнений (СЛАУ) в составе неявного метода интегрирования по времени программного комплекса NOISEtte [1]. Этот комплекс предназначен для моделирования задач газовой динамики на гибридных суперкомпьютерах. Для дискретизации по пространству используются схемы повышенной точности [3] на неструктурированных смешанных сетках. Интегрирование по времени осуществляется с помощью неявных схем на основе линеаризации по Ньютону (BDF1, BDF2). Параллельный алгоритм и гетерогенная реализация для центральных (CPU) и графических (GPU) процессоров представлена в [1]. Для многоуровневого распараллеливания используются стандарты MPI, OpenMP и OpenCL. Открытый стандарт OpenCL позволяет задействовать GPU различной архитектуры, включая NVIDIA, AMD, Intel. Также код адаптирован к отечественной архитектуре Эльбрус [2].

В настоящее время значительно возросла актуальность импортозамещения программного обеспечения (ПО) для промышленного суперкомпьютерного моделирования. В вихреразрешающем моделировании турбулентности, для которого был предназначен NOISEtte, величина шага по времени в неявной схеме естественным образом ограничена динамикой моделируемых структур течения. Поэтому СЛАУ с матрицей Якоби как правило не требует сложных решателей: на практике было достаточно итерационного метода Bi-CGSTAB [4] с простейшим предобуславливателем на основе блочного метода Якоби. Однако в промышленных задачах зачастую используется методы на основе осредненных по Рейнольдсу уравнений Навье-Стокса, допускающие намного больший шаг по времени, что делает СЛАУ более

[1]Студент факультета ВМК МГУ имени М.В. Ломоносова

[2]Доцент ВМК МГУ имени М.В. Ломоносова, вед. научн. сотр. ИПМ имени М.В. Келдыша РАН

*Работа выполнена при финансовой поддержке РНФ, проект 19-11-00299.

жесткой и требует от решателя выполнения большего числа итераций. Нужны предобуславливатели, более существенно улучшающие сходимость, но с ними есть проблема: матрица меняется на каждом шаге по времени, из-за чего методы с вычислительно дорогим этапом построения (например, на основе ресурсоемкой LU факторизации) оказываются неэффективны или неприменимы. Другой проблемой является крайне ограниченный объем памяти GPU, что также накладывает существенные ограничения на выбор метода. В данной работе применяется многоцветный симметричный метод Гаусса–Зейделя, у которого затраты памяти и вычислительная стоимость построения незначительны и сопоставимы с таковыми у метода Якоби.

Описание параллельного алгоритма

Особенностью матрицы Якоби в неявной схеме интегрирования по времени для уравнений Навье–Стокса является ее блочная структура. Размер блока основной СЛАУ равен 5 – по числу физических или консервативных переменных. Портрет блочной матрицы соответствует смежности сеточных узлов по ребрам. При использовании модели турбулентности возникает еще одна или несколько дополнительных матриц для турбулентных переменных. Решатель СЛАУ работает сразу с несколькими матрицами и наборами векторов – для основной и дополнительных систем. Поскольку матрицы имеют общий портрет, у дополнительных матриц хранятся только коэффициенты. Матрицы хранятся в блочном построчно-разреженном формате CSR (Compressed Sparse Row). Для работы решателя реализован набор кернел-подпрограмм для базовых операций линейной алгебры, включающий матрично-векторное произведение – SpMV (Sparse Matrix-Vector), линейную комбинацию двух векторов ($\mathbf{x} = a\mathbf{y} + b\mathbf{y}$), скалярное произведение векторов. Для снижения накладных расходов на обмен данными, который необходим, в частности, для матрично-векторного произведения, SpMV, используется сокрытие обменов за вычислениями [1].

Симметричный метод Гаусса–Зейделя, использующийся в качестве предобуславливателя для метода Bi-CGSTAB [4], работает следующим образом. Решается СЛАУ $\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{A} \in \mathbb{R}^{n\times n}$. Матрица $\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}$ представляется в виде суммы нижней треугольной части, диагональной и верхней треугольной части, соответственно. Итерация метода состоит из двух этапов – решение СЛАУ с нижнетреугольной (прямая подстановка) и с верхнетреугольной матрицей (обратная подстановка):

$$(\mathbf{L}+\mathbf{D})\mathbf{x}^{k+1/2} = \mathbf{b} - \mathbf{U}\mathbf{x}^{k}; \quad (\mathbf{D}+\mathbf{U})\mathbf{x}^{k+1} = \mathbf{b} - \mathbf{L}\mathbf{x}^{k+1/2}.$$

Прямая (и аналогичная обратная) подстановка имеет зависимости по данным, препятствующие распараллеливанию на GPU:

$$x_i^{k+1/2} = 1/a_{ii} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1/2} - \sum_{j=i+1}^{n} a_{ij} x_j^k \right).$$

Для нахождения i-й позиции вектора требуется знать все предыдущие значения. Чтобы разорвать эти зависимости используется традиционный в таких случаях подход с переупорядочиванием блоков неизвестных на основе раскраски графа, портрет матрицы смежности которого соответствует портрету блочной матрицы решаемой системы за вычетом главной диагонали. Раскраска широко применяется для реализации алгоритмов на основе метода Гаусса–Зейделя на GPU, см., например, [5,6].

Для раскраски вершин графа в данной работе используется простейший жадный алгоритм. Блоки неизвестных, соответствующие вершинам графа, переупорядочиваются по цветам, в результате чего в прямой и обратной подстановке отсутствуют неизвестные того же цвета с нового итерационного слоя. После переупорядочивания алгоритм полностью совместим с параллельной парадигмой потоковой обработки, применяющейся на GPU: блоки переменных одного цвета идут подряд; наборы блоков, сгруппированные по цветам, обрабатываются по очереди; каждый цвет обрабатывается в параллельном режиме. Алгоритм раскраски применяется однократно при запуске расчета и не вносит заметного вклада в общее время вычислений. Этап построения предобуславливателя отличается низкой ресурсоемкостью. Как и в методе Якоби, выполняется только обращение диагональных блоков матрицы, для хранения обратных блоков выделяется дополнительная память.

Гетерогенная реализация для CPU и GPU

Многоцветный параллельный метод Гаусса–Зейделя реализуется для CPU тривиально средствами параллелизма циклов OpenMP. После переупорядочивания для каждого цвета получается непрерывный диапазон индексов блоков неизвестных. Модификация прямой и обратной подстановки состоит в добавлении внешнего цикла по цветам, сужении диапазона цикла подстановки до одного цвета и добавлении циклической директивы OpenMP этому циклу. Однако важно отметить, что переупорядочивание по цветам может заметно ухудшать сходимость итерационного решателя. Поэтому в данной работе вместо многоцветного варианта на CPU используется более эффективная версия – на основе декомпозиции. Похожий алгоритм применен, например, в работе [7].

В версии для CPU неизвестные распределяются по подобластям MPI процессов и OpenMP потоков путем декомпозиции сеточного графа с минимизацией разрезанных ребер. Неизвестные упорядочиваются по подобластям, а внутри подобластей используется нумерация по алгоритму

Катхилла–Макки для повышения локальности доступа к данным в памяти.

Множественные подобласти обрабатываются одновременно. Внутри подобластей используется метод Гаусса–Зейделя, а на интерфейсах между подобластями происходит, по сути, переключение на метод Якоби (т.е. значения из позиций вектора решения, принадлежащих другим подобластям, берутся для прямой подстановки с предыдущей итерации, а для обратной подстановки из результатов прямой подстановки). Такой подход дает лишь незначительную деградацию сходимости по сравнению последовательной версией, но плохо подходит для GPU.

В реализации многоцветного метода для GPU вместо параллельного цикла подстановки используется вызов кернел-функции OpenCL (кернелами в терминологии OpenCL называются подпрограммы, выполняющиеся на ускорителе), обрабатывающей неизвестные одного цвета на GPU в потоковом режиме. Соответственно, одна итерация симметричного метода требует по два вызова кернела OpenCL на каждый цвет – для прямой и обратной подстановки. Простейшая реализация на OpenCL состоит в переносе из CPU версии многоцветного метода тела цикла по блокам неизвестных одного цвета напрямую в кернел-функцию. Таким образом, каждая итерация цикла становится рабочей единицей, т.е. элементарным заданием, распределяемым при потоковой обработке. Вместо переменной счетчика цикла в кернел-функции имеется идентификатор рабочей единицы.

К сожалению, такая простая реализация не показала существенного ускорения на GPU относительно CPU. Для повышения производительности GPU версии пришлось многократно уменьшить задание рабочей единицы. В исходной версии одна рабочая единица кернела OpenCL обрабатывает один блок из 5 неизвестных, оперируя матричными блоками из 25 коэффициентов, соответственно. В оптимизированной версии один блок обрабатывается сразу 25 рабочими единицами, одна единица отвечает за лишь за один коэффициент в матричных блоках. Редукция сумм в 5 позиций выходного вектора осуществляется с использованием быстрой локальной памяти мультипроцессора. Размер рабочей группы для GPU NVIDIA равен 128, одна группа обрабатывает 5 блоков, 3 единицы в группе остаются неактивны. За счет такого уменьшения элементарных заданий сокращается потребление регистровой памяти в пересчете на рабочую единицу, что позволяет достигать намного более высокой загрузки исполнительных устройств потокового мультипроцессора GPU.

Производительность на CPU и GPU

Производительность исследована на репрезентативной задаче, в которой моделируется двухлопастной несущий винт вертолета (Рис. 1).

Рассматривается одна лопасть в условиях периодичности, сетка смешанная, неструктурированная, преимущественно тетраэдральная (80%), состоит из 1.3 млн узлов. Используется неявная схема 1 порядка, число Куранта около 10^3. Критерий остановки решателя СЛАУ – невязка в L2 норме относительно нормы правой части $\varepsilon = 0.01$. Гибридный вычислительный сервер для тестирования имеет два 16-ядерных CPU Intel Xeon Gold 5218 2.30GHz (2 потока на ядро, 6 каналов DDR4-2666 – 128ГБ/с); GPU NVIDIA A5000 (24 GB GDDR6, 768 ГБ/с).

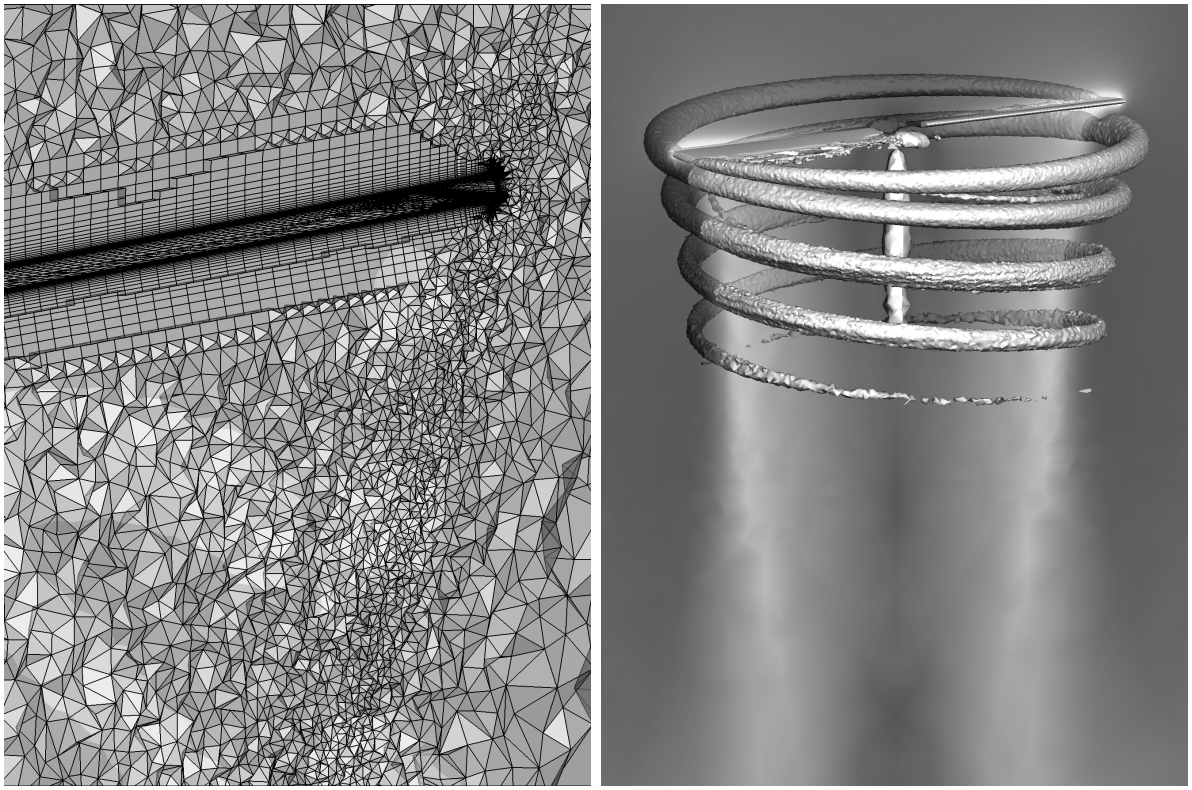

Рис. 1. Тестовая задача. Фрагмент неструктурированной смешанно-элементной сетки (слева) и визуализация концевого вихря (справа).

Поскольку переупорядочивание по цветам может ухудшать сходимость, при демонстрации ускорения на GPU относительно CPU, естественно, на CPU используется более эффективная версия на основе декомпозиции. На каждой итерации Bi-CGSTAB присутствует по два вызова предобуславливателя, в котором выполняется две итерации симметричного метода Гаусса–Зейделя. В данном тесте среднее число итераций Bi-CGSTAB для CPU версии предобуславливателя – 6, для GPU версии – 10, т. е. GPU выполняет больший объем работы из-за ухудшения сходимости. Среднее количество ненулевых блоков в строках матрицы – 12.1, количество цветов в раскраске – 11, размер блока – 5, общее количество неизвестных – 6.8 млн.

Результаты представлены в таблице 1 для трех версий: версия для центральных процессоров (обозначена CPU), простейшая версия для GPU (GPU-Base), оптимизированная версия для GPU (GPU-Opt). В таблице

Табл. 1. Производительность многоцветного метода Гаусса–Зейделя

Версия	Ресурсы	MPI	OpenMP	Время, с	Ускорение	GFLOPS	ГБ/с
CPU	1 CPU	1	1	29.5	1	1.3	5.4
CPU	1 CPU	1	32	3.9	7.6	10	42
CPU	2 CPU	2	32	2.2	13.4	17.7	74
GPU-Base	1 GPU	1	32	3.6	8.2	18	46
GPU-Opt	1 GPU	1	32	0.52	57	125	519
CPU, GPU-Opt	1 CPU, 1 GPU	2	28, 4	0.48	61.5	135	563
GPU-Opt	2 GPU	2	32	0.3	98	216	900

приведены задействованные ресурсы; количество MPI процессов; число OpenMP нитей на процесс (если различается, указано через запятую для всех процессов); затраты времени на предобуславливатель при решении СЛАУ, осредненные по 10 шагам по времени; ускорение относительно последовательного режима на CPU; производительность в операциях с плавающей точкой в секунду (FLOPS); оценка снизу по использованию пропускной способности памяти.

Заключение

В результате работы создана гетерогенная реализация предобуславливателя, позволяющая задействовать множественные CPU и GPU. Из таблицы 1 можно сделать следующие выводы. Применение раскраски для симметричного метода Гаусса – Зейделя позволяет получать значительное ускорение на GPU относительно CPU даже несмотря на некоторое ухудшение сходимости из-за переупорядочивания по цветам. Для данных моделей устройств ускорение на GPU относительно 16-ядерного CPU составило 7.5 раз, что дает практический эквивалент с одного GPU примерно 120 ядер. Стоит отметить, что модель GPU NVIDIA A5000 является сравнительно бюджетной, примерно в 3 раза дешевле широко распространенных в настоящее время вычислительных GPU NVIDIA V100.

Гетерогенный режим с одновременным использованием CPU и GPU дает определенный выигрыш, но при таком соотношении производительности не имеет большого смысла: потенциал ускорения от использования CPU лишь 13%. По факту получается менее 8% из-за того, что добавляются потери на обмен данными и неизбежные погрешности

балансировки, поскольку подобласти CPU и GPU балансируются по соотношению производительности всего расчетного алгоритма, а не отдельно предобуславливателя.

Основной же вывод состоит в том, что при работе с такими мелкоблочными матрицами крайне важна оптимизация OpenCL кернел-функций под архитектуру потокового мультипроцессора GPU, которая заключается в минимизации заданий рабочих единиц и подборе размеров рабочих групп. При соответствующей оптимизации GPU весьма эффективно справляется с таким вычислительным алгоритмом, скорость обработки данных близка к 70% от пропускной способности памяти, другими словами, получаемая производительность составляет около 70% от теоретически достижимой на данном устройстве для данного алгоритма.

Благодарности. Работа выполнена с использованием оборудования Центра коллективного пользования сверхвысокопроизводительными вычислительными ресурсами МГУ имени М. В. Ломоносова. Работа выполнялась с использованием инфраструктуры Центра коллективного пользования «Высокопроизводительные вычисления и большие данные» (ЦКП «Информатика») ФИЦ ИУ РАН (г. Москва). Вычисления проведены с помощью гибридного вычислительного модуля, установленного в Суперкомпьютерном Центре коллективного пользования ИПМ им. М. В. Келдыша РАН.

Литература

1. *Gorobets A., Bakhvalov P.* Heterogeneous CPU+GPU parallelization for high-accuracy scale-resolving simulations of compressible turbulent flows on hybrid supercomputers // Computer Physics Communications. 2022. Vol. 271, 108231. https://doi.org/10.1016/j.cpc.2021.108231

2. *Горобец А.В., Нейман-заде М.И., Окунев С.К., Калякин А.А., Суков С.А.* Производительность отечественного процессора Эльбрус-8С в суперкомпьютерном моделировании задач вычислительной газовой динамики // Математическое моделирование. 2019. Том 31. № 44. С. 17–32.

3. *Bakhvalov P., Kozubskaya T.* EBR-WENO scheme for solving gas dynamics problems with discontinuities on unstructured meshes // Computers & Fluids. 2017. Vol. 157, P. 312–324. https://doi.org/10.1016/j.compfluid.2017.09.004

4. *Van der Vorst H. A.* Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems // SIAM Journal on Scientific Computing. 1992. Vol. 13, P. 631–644. https://doi.org/10.1137/0913035.

5. Phillips E., Fatica M. A CUDA Implementation of the High Performance Conjugate Gradient Benchmark // PMBS 2014. Lecture Notes in Computer Science, Springer, Cham. 2015. Vol. 8966, P. 68–84. https://doi.org/10.1007/978-3-319-17248-4_4

6. *Menshov I., Pavlukhin P.* Highly scalable implementation of an implicit matrix-free solver for gas dynamics on GPU-accelerated clusters // The Journal of Supercomputing. 2017. Vol. 73, P. 631–638. https://doi.org/10.1007/s11227-016-1800-1

7. *Petrov M.N., Titarev V.A., Utyuzhnikov S.V., Chikitkin A.V.* A multithreaded OpenMP implementation of the LU-SGS method using the multilevel decomposition of the unstructured computational mesh // Computational Mathematics and Mathematical Physics. 2017. Vol. 57, no. 11, P. 1856–1865. https://doi.org/10.1134/S0965542517110124